This document formally defines the simply typed, call-by-name λ-calculus and proves it sound.

1 Syntax

The metavariable \( I \) ranges over an infinite set of variable names. The metavariable \( E \) ranges over expressions (terms). The metavariable \( \tau \) ranges over types. The metavariable \( V \) ranges over values.

\[
E ::= I \mid \lambda I : \tau. E \mid E_1 E_2
\]

\[
\tau ::= \ast \mid \tau_1 \rightarrow \tau_2
\]

\[
V ::= \lambda I : \tau. E
\]

2 Static Semantics

The metavariable \( \Gamma \) represents a type environment, which is a set of \( (I:\tau) \) pairs. A type environment has at most one pair for a given variable name; this can always be ensured via renaming of bound variables. We extend a type environment with additional pairs using the \( \cup \) operator, which yields the union of its argument sets of pairs if those sets have disjoint variable names, and is undefined otherwise. We use \( \emptyset \) to denote the empty type environment.

A judgment of the form \( \Gamma \vdash E : \tau \) means “expression \( E \) has type \( \tau \) under the typing assumptions in \( \Gamma \).”

\[
\frac{I : \tau \in \Gamma}{\Gamma \vdash I : \tau} \quad \text{(T-Var)}
\]

\[
\frac{\Gamma \cup \{I : \tau\} \vdash E : \tau'}{\Gamma \vdash (\lambda I : \tau. E) : \tau \rightarrow \tau'} \quad \text{(T-λ)}
\]

\[
\frac{\Gamma \vdash E_1 : \tau \rightarrow \tau' \quad \Gamma \vdash E_2 : \tau}{\Gamma \vdash E_1 E_2 : \tau'} \quad \text{(T-App)}
\]

3 Dynamic Semantics

3.1 Substitution

The substitution function, written \([E_2/I]E_1\) and meaning “replace all free occurrences of \( I \) in \( E_1 \) with \( E_2 \), avoiding capture,” is defined below. We assume that renaming of bound variables is applied as necessary to make the side
conditions of the third case hold.

\[
\begin{align*}
[E_2/I]I & = E_2 \\
[E_2/I]J & = J & \text{if } J \neq I \\
[E_2/I](\lambda J : \tau. E_1) & = \lambda J : \tau. [E_2/I]E_1 & \text{if } J \neq I \text{ and } J \notin \text{FV}(E_2) \\
[E_2/I](E_1 E_2) & = ([E_2/I]E_1)([E_2/I]E_2)
\end{align*}
\]

3.2 Evaluation Rules

The judgment \( E \to E' \) means “expression \( E \) evaluates in one step to \( E' \).”

\[
\frac{(\lambda I : \tau. E_1)E_2 \to [E_2/I]E_1}{E_1 \to E'_1} \quad \text{(E-App1)} \\
\frac{E_1 \to E'_1}{E_1 E_2 \to E'_1 E_2} \quad \text{(E-App2)}
\]

4 Type Soundness

4.1 Progress

Lemma (Canonical Forms):

a. If \( \emptyset \vdash V : \tau_1 \to \tau_2 \) then \( V \) has the form \( \lambda I : \tau_1. E \).

Proof: Immediate from rule T-\( \lambda \) and the fact that no other typing rules apply to a value of type \( \tau_1 \to \tau_2 \).

Theorem (Progress): If \( \emptyset \vdash E : \tau \), then either \( E \) is a value or there exists \( E' \) such that \( E \to E' \).

Proof: By induction on the typing derivation of \( \emptyset \vdash E : \tau \).

We proceed via a case analysis of the last rule in the derivation:

- Case T-Var: Then \( E = I \) and \( I : \tau \in \emptyset \).
  This is a contradiction, and so T-Var cannot be the last rule in the derivation.

- Case T-\( \lambda \): Then \( E = \lambda I : \tau_1. E_1 \).
  \( E \) is a value.

- Case T-App: Then \( E = E_1 E_2 \) and \( \emptyset \vdash E_1 : \tau_2 \to \tau \) and \( \emptyset \vdash E_2 : \tau_2 \).
  By the inductive hypothesis, either \( E_1 \) is a value or there exists \( E'_1 \) such that \( E_1 \to E'_1 \).

  We perform a case analysis on these two possibilities:

  - Case there exists \( E'_1 \) such that \( E_1 \to E'_1 \):
    By E-App2, \( E_1 E_2 \to E'_1 E_2 \).
    Thus \( E' = E_1 E_2 \).

  - Case \( E_1 \) is a value \( V_1 \):
    Since \( \emptyset \vdash V_1 : \tau_2 \to \tau \), by the Canonical Forms lemma, \( V_1 \) has the form \( \lambda I : \tau_2. E_3 \).
    By E-App1, \( (\lambda I : \tau_2. E_3)E_2 \to [E_2/I]E_3 \).
    Thus \( E' = [E_2/I]E_3 \).
4.2 Preservation

Lemma (Permutation): If $\Gamma \triangleright \{ I_1 : \tau_1 \} \cup \{ I_2 : \tau_2 \} \vdash E : \tau$, then $\Gamma \triangleright \{ I_2 : \tau_2 \} \cup \{ I_1 : \tau_1 \} \vdash E : \tau$.

Proof: By the fact that $\triangleright$ is a commutative operator.

Lemma (Weakening): If $\Gamma \vdash E : \tau$ and $I' \notin \text{dom}(\Gamma)$, then $\Gamma \triangleright \{ I' : \tau' \} \vdash E : \tau$.

Proof: By induction on the typing derivation of $\Gamma \vdash E : \tau$.

We proceed via a case analysis of the last rule in the derivation:

- Case T-Var: Then $E = I$ and $I : \tau \in \Gamma$.
  - Since $I' \notin \text{dom}(\Gamma)$, we know $I \neq I'$ and so $\Gamma \triangleright \{ I' : \tau' \}$ is defined.
  - Therefore $I : \tau \in \Gamma \triangleright \{ I' : \tau' \}$.
  - By T-Var, $\Gamma \triangleright \{ I' : \tau' \} \vdash I : \tau$.

- Case T-\lambda: Then $E = \lambda I_1 : \tau_1. E_2$ and $\tau = \tau_1 \rightarrow \tau_2$ and $\Gamma \triangleright \{ I_1 : \tau_1 \} \vdash E_2 : \tau_2$.
  - We assume w.l.o.g. that $I_1 \neq I'$, renaming $I_1$ if necessary.
  - Since $I' \notin \text{dom}(\Gamma)$ and $I_1 \neq I'$, then $I' \notin \text{dom}(\Gamma \cup \{ I_1 : \tau_1 \})$.
  - By the inductive hypothesis, $\Gamma \cup \{ I_1 : \tau_1 \} \triangleright \{ I' : \tau' \} \vdash E_2 : \tau_2$.
  - By Permutation, $\Gamma \triangleright \{ I' : \tau' \} \cup \{ I_1 : \tau_1 \} \vdash E_2 : \tau_2$.
  - By T-\lambda, $\Gamma \triangleright \{ I' : \tau' \} \vdash (\lambda I_1 : \tau_1. E_2) : \tau_1 \rightarrow \tau_2$.

- Case T-App: Then $E = E_1 E_2$ and $\Gamma \vdash E_1 : \tau_2 \rightarrow \tau$ and $\Gamma \vdash E_2 : \tau_2$.
  - By the inductive hypothesis, $\Gamma \triangleright \{ I' : \tau' \} \vdash E_1 : \tau_2 \rightarrow \tau$ and $\Gamma \triangleright \{ I' : \tau' \} \vdash E_2 : \tau_2$.
  - By T-App, $\Gamma \triangleright \{ I' : \tau' \} \vdash E_1 E_2 : \tau$.

Corollary: If $\Gamma \vdash E : \tau$ and $\Gamma \triangleright \Gamma'$ is defined, then $\Gamma \triangleright \Gamma' \vdash E : \tau$.

Proof: By repeated applications of Weakening.

Lemma (Substitution Preserves Typing): If $\Gamma \triangleright \{ I_2 : \tau_2 \} \vdash E_1 : \tau_1$ and $\emptyset \vdash E_2 : \tau_2$, then $\Gamma \triangleright [E_2/I_2]E_1 : \tau_1$.

Proof: By induction on the typing derivation of $\Gamma \triangleright \{ I_2 : \tau_2 \} \vdash E_1 : \tau_1$.

We proceed via a case analysis of the last rule in the derivation:

- Case T-Var: Then $E_1 = I_1$ and $I_1 : \tau_1 \in \Gamma \triangleright \{ I_2 : \tau_2 \}$.
  - There are two subcases to consider, depending on whether or not $I_1 = I_2$:
    - Case $I_1 = I_2$: Then $[E_2/I_2]I_1 = [E_2/I_1]I_1 = E_2$, and so we need to show $\Gamma \vdash E_2 : \tau_1$.
      - By definition of $\triangleright$, $I_2 \notin \text{dom}(\Gamma)$.
      - Since $I_2 : \tau_1 \in \Gamma \triangleright \{ I_2 : \tau_2 \}$ and $I_2 \notin \text{dom}(\Gamma)$, $I_2 : \tau_1 \in \{ I_2 : \tau_2 \}$ and so $\tau_1 = \tau_2$.
      - Since $\emptyset \vdash E_2 : \tau_2$, by Weakening $\Gamma \vdash E_2 : \tau_1$.
    - Case $I_1 \neq I_2$: Then $[E_2/I_2]I_1 = I_1$, and so we need to show $\Gamma \vdash I_1 : \tau_1$.
      - Since $I_1 : \tau_1 \in \Gamma \triangleright \{ I_2 : \tau_2 \}$ and $I_1 \neq I_2$, we know $I_1 : \tau_1 \in \Gamma$.
      - By T-Var, $\Gamma \vdash I_1 : \tau_1$.

- Case T-\lambda: Then $E_1 = \lambda I_0 : \tau_0. E'_1$ and $\tau_1 = \tau_0 \rightarrow \tau'_1$ and $\Gamma \triangleright \{ I_2 : \tau_2 \} \cup \{ I_0 : \tau_0 \} \vdash E'_1 : \tau'_1$.
  - Then $[E_2/I_2](\lambda I_0 : \tau_0. E'_1) = \lambda I_0 : \tau_0. [E_2/I_2]E'_1$, where $I_0 \neq I_2$ and $I_0 \notin \text{FV}(E_2)$, which we can assume w.l.o.g. by renaming $I_0$ appropriately. So we need to show $\Gamma \vdash (\lambda I_0 : \tau_0. [E_2/I_2]E'_1) : \tau_0 \rightarrow \tau'_1$.
    - By Permutation, $\Gamma \triangleright \{ I_0 : \tau_0 \} \cup \{ I_2 : \tau_2 \} \vdash E'_1 : \tau'_1$.
    - By the inductive hypothesis, $\Gamma \triangleright \{ I_0 : \tau_0 \} \vdash [E_2/I_2]E'_1 : \tau'_1$.
    - By T-\lambda, $\Gamma \vdash (\lambda I_0 : \tau_0. [E_2/I_2]E'_1) : \tau_0 \rightarrow \tau'_1$.
• Case T-App: Then $E_1 = E'_1, E''_1$ and $\Gamma \ni \{I_2 : \tau_2\} \vdash E'_1 : \tau''_1 \rightarrow \tau_1$ and $\Gamma \ni \{I_2 : \tau_2\} \vdash E''_1 : \tau''_1$.

Then $[E_2/I_2](E'_1 E''_1) = ([E_2/I_2]E'_1) ([E_2/I_2]E''_1)$, so we need to show $\Gamma \vdash (([E_2/I_2]E'_1) ([E_2/I_2]E''_1)) : \tau_1$.

By the inductive hypothesis, $\Gamma \vdash [E_2/I_2]E'_1 : \tau''_1 \rightarrow \tau_1$ and $\Gamma \vdash [E_2/I_2]E''_1 : \tau''_1$.

By T-App, $\Gamma \vdash ([E_2/I_2]E'_1) ([E_2/I_2]E''_1)) : \tau_1$.

**Theorem** (Preservation): If $\emptyset \vdash E : \tau$ and $E \rightarrow E'$, then $\emptyset \vdash E' : \tau$.

**Proof:** By induction on the typing derivation of $\emptyset \vdash E : \tau$.

We proceed via a case analysis of the last rule in the derivation:

• Case T-Var: Then $E = I$.

But by inspection of the operational semantics, there is no $E'$ such that $I \rightarrow E'$, so this is a contradiction, and so T-Var cannot be the last rule in the derivation.

• Case T-lam: Then $E = \lambda I : \tau_1. E_1$.

But by inspection of the operational semantics, there is no $E'$ such that $\lambda I : \tau_1. E_1 \rightarrow E'$, so this is a contradiction, and so T-\lam cannot be the last rule in the derivation.

• Case T-App: Then $E = E_1 E_2$ and $\emptyset \vdash E_1 : \tau_2 \rightarrow \tau$ and $\emptyset \vdash E_2 : \tau_2$.

We’re given that $E_1 E_2 \rightarrow E'$. We proceed by a case analysis on the last rule used in the derivation of this reduction step:

– Case E-App2: Then $E' = E'_1 E_2$ and $E_1 \rightarrow E'_1$.

By the inductive hypothesis, $\emptyset \vdash E'_1 : \tau_2 \rightarrow \tau$.

By T-App, $\emptyset \vdash E'_1 E_2 : \tau$.

– Case E-App1: Then $E_1 = \lambda I : \tau'. E_3$ and $E' = [E_2/I]E_3$.

Since $\emptyset \vdash (\lambda I : \tau'. E_3) : \tau_2 \rightarrow \tau$, by inspection of the typing rules, T-lam must have been the typing rule applied to prove this judgment, and so we know $\tau' = \tau_2$ and the rule’s premise, $\emptyset \ni \{I : \tau_2\} \vdash E_3 : \tau$.

By the Substitution lemma, $\emptyset \vdash [E_2/I]E_3 : \tau$.

### 4.3 Soundness

**Theorem** (Soundness): If $\emptyset \vdash E : \tau$ then either $E$ is a value or there exists $E'$ such that $E \rightarrow E'$ and $\emptyset \vdash E' : \tau$.

**Proof:** Since $\emptyset \vdash E : \tau$, by Progress either $E$ is a value or there exists $E'$ such that $E \rightarrow E'$. In the latter case, by Preservation we have $\emptyset \vdash E' : \tau$. 4